UNIVERSITÉ DE NICE – SOPHIA ANTIPOLIS
UFR FACULTÉ DES SCIENCES
Parc Valrose – BP 71 – F 06108 NICE Cedex 2

Licence MASS 2e année
Traitements de données textuelles 2007-2008
Pierre Crescenzo, Jean-Pierre Lips et Nicolas Nobelis
inspirés par Roger Rousseau

Séance de TP n° 08 de la semaine du 29 octobre 2007

Scripts Shell : Gestion d'une corbeille

Mise en place

Créez votre répertoire de séance `s08` sous `~/tp` et placez-vous dedans. Il n'y a pas d'archive de fourniture pour cette séance.

Objectifs : gestion d'une corbeille

Il s'agit d'écrire un ensemble de scripts pour gérer les effacements de **fichiers ordinaires non cachés** avec une corbeille (à peu près comme dans Windows ou l'environnement Gnome de Linux). Le principe consiste à effacer les fichiers à l'aide d'une commande spéciale **`effacer`** qui *ne les efface pas vraiment*, mais les déplace dans un répertoire appelé « *la corbeille* ».
Pour éviter de spécifier la corbeille utilisée en argument de chaque commande, vous utiliserez une variable d'environnement **`CORBEILLE`** dont la valeur contiendra le nom du répertoire spécifique.
Les premiers scripts seront très simples et ne feront aucun contrôle, en supposant que tout va bien. Les scripts suivants seront des versions améliorées qui testeront les cas d'erreurs et fourniront quelques options.

1. Gestion simplifiée de la corbeille

1.1. *Initialisations*

Ecrivez un script **`initialiser`**, qui initialise le fonctionnement de la corbeille de la manière suivante :

- Définit et exporte une variable **`CORBEILLE`** dont la valeur est le nom du répertoire qui va servir de corbeille, en l'occurrence **`~/Corbeille`** .
- Crée un répertoire pour la corbeille, s'il n'existe pas déjà. S'il existe déjà, supprime les fichiers qu'il contient.
- Affiche la valeur de la variable **`CORBEILLE`**.
- Liste le contenu de la corbeille en utilisant la commande **`ls`** et la variable **`CORBEILLE`**.

Pour rendre effectives les initialisations de variables, ce script sera appelé dans le shell courant, sans créer de nouveau processus.

1.2. *Effacement simplifié d'un fichier*

Écrivez un script **`effacer`** qui implémente la spécification suivante :

- Syntaxe : `effacer fichier`
- Effet : efface le fichier *fichier* en gardant une copie dans le répertoire de la corbeille. L'emplacement de ce répertoire est donné dans la variable d'environnement **`CORBEILLE`**.

Créez puis effacer des fichiers pour tester ce script. Listez le contenu de la corbeille pour vérifier son bon fonctionnement.

1.3. *Vidage simplifié de la corbeille*

Écrivez un script **`vider`** qui ne prend aucun paramètre, mais utilise la variable **`CORBEILLE`** pour effacer **complètement** les fichiers se trouvant dans la corbeille.

2. Commandes robustes

2.1. *Vidage robuste de la corbeille*

Écrivez une autre commande **vider2** qui fonctionne comme **vider**, mais de manière plus robuste. Cette version commence par tester :

- si la variable **CORBEILLE** existe,
- si le répertoire de corbeille existe,
- et finalement si l'écriture et la traversée de ce répertoire sont permis.

Si un de ces tests échoue, cette version arrête l'exécution immédiatement avec un code de retour **1** et un message d'erreur sur la *voie standard d'erreur*.

Dans le cas contraire, **vider2** affiche sur la *voie standard d'erreur* un message d'avertissement si la corbeille est vide et sinon un message donnant le nombre de fichiers effectivement supprimés.

```
# exemple de vidage de la corbeille
> vider2
8 fichiers supprimes
> vider2
La corbeille est deja vide...
```

2.2. *Effacement robuste de fichiers*

Ecrivez une nouvelle version **effacer2** de la commande **effacer** à partir de la spécification suivante. **Lire le sujet de cet exercice en entier, y compris les remarques finales, avant de commencer.**

- Syntaxe : `effacer2 fichier...`
- Effet : efface **les** fichiers donnés en argument, en gardant une copie dans le répertoire de la corbeille. L'emplacement de ce répertoire est donné dans la variable d'environnement **CORBEILLE**.
- Robustesse : avant de faire son travail, la commande doit tester :
  - ◇ qu'il y a au moins un paramètre transmis, sinon elle affichera le mode d'emploi sur la voie standard d'erreur et quittera immédiatement avec un code de retour 1 . Le mode d'emploi sera stocké dans une variable interne **USAGE**.
  - ◇ si la variable **CORBEILLE** existe, sinon la valeur par défaut **~/Corbeille** sera utilisée.
  - ◇ si le répertoire de la corbeille existe et permet l'écriture et la traversée. Sinon, il sera créé avec les permissions adéquates.
  - ◇ si chacun des fichiers donnés en paramètre peut être déplacé. Pour cela, on construira deux listes : celle des fichiers qui existent (qui vont donc pouvoir effectivement être déplacés dans la corbeille) et la liste des fichiers erronés (qui n'existent pas). Ces deux listes seront représentées par des chaînes contenant les noms des fichiers séparés par un espace.

  En l'absence d'erreur, la commande renverra après son travail le code de retour **0** indiquant un succès.

```
# on suppose dans cet exemple que f1, f2 et f3 existent
> effacer2 f1 titi f2 f3 toto tata
fichiers existants : f1 f2 f3
fichiers inexistants : titi toto tata
```

Pour écrire ce script, on procédera de **manière incrémentale** : on partira de la version non robuste, sans contrôle. Puis on ajoutera progressivement les différents contrôles **en testant à chaque fois le contrôle avant de passer** au contrôle suivant.
Une dernière indication : pour manipuler les fichiers indiqués dans une chaîne de noms séparés par un espace, il faut dire au Shell d'effectuer une analyse de la commande après avoir effectué toutes ses transformations ou substitutions. Cela s'obtient avec la commande **eval**, par exemple :

```
> ls "/tmp ~"
ls: /tmp ~: Aucun fichier ou repertoire de ce type
> eval ls "/tmp ~"
/tmp:
 ...
```

3. Pour les plus courageux ...

3.1. *Vidage interactif de la corbeille*

Écrire une version **vider3** qui part de **vider2** et accepte en plus l'option *-i* pour un comportement interactif qui demande confirmation avant l'effacement de chaque fichier de la corbeille. Pour afficher une demande de confirmation, on utilisera la commande **echo** avec l'option **-n**, ce qui évite de passer à la ligne pour lire la réponse. Pour lire la réponse, on utilisera la commande **read REP**, qui lit une ligne sur *stdin* et copie son contenu dans la variable **REP**.